

Vladan Djerić
991573659

Project Proposal

I am proposing an extension to the application replay project described in “Correlating Multi-Session Attacks via Replay” [1] by adding automated network replay capabilities. This project’s larger goal is to improve the accuracy of dependency generation with the “Taser” intrusion recovery system.

MOTIVATION

Currently, intrusion analysis and recovery is a manual and tedious process. There have been several attempts at expediting the operation by automating some aspects of the process. Taser intrusion recovery is one such system. It logs system calls and is able to derive a dependency tree of kernel objects which aids in determining the list of files tainted by an intruder or software misconfiguration. Additionally, through the use of logging and snapshots, Taser is able to restore the tainted files to their pre-intrusion versions. Although Taser presents a powerful new approach to intrusion analysis and recovery, it suffers from systematic false positives in certain scenarios and is vulnerable to multi-session attacks. For example, if a heavily shared file is modified by a tainted object, every process that reads the shared file will, in turn, become marked as tainted by Taser. In many cases, such as a password file, this will be inappropriate. An approach based on application replay has been proposed to aid in establishing whether a file is truly tainted. The application would be re-executed several times in the same environment, using tainted inputs (such as files) and clean versions of the inputs. If the output of the application is significantly different with the untainted version of the input, for example the attacker is unable to log in, we can conjecture that the application was dependent on the tainted input. This could be an effective technique for eliminating false positives.

IMPLEMENTATION

The replay project proposal has already been published in [1]. The rest of this proposal deals with the implementation of network replay, which I will undertake, separate from the development of other replay infrastructure. I will implement the networking portion of the project in co-operation with Wei Zhang, a student in the Dependable Software class.

For our purposes, the replay framework will consist of 3 modules: the application under test, a replay module which intercepts system calls, and a network proxy module charged with replaying network traffic. The replay module uses ptrace to intercept system calls and to directly read and write process memory. The ability to access memory is used to carry out logging and to rewrite networking system calls so that they are redirected to the network proxy module. It should be noted that the proxy and logger both operate in user-mode and that no modifications to the kernel are necessary for this prototype of network replay. Since the interception is carried out at the system call boundary, only application

layer network data is captured and the system is unconcerned with lower layers. The network proxy corrects protocol fields which change between replays (such as cookie fields or checksums) by relying on protocol-specific knowledge. For protocols for which it does not contain protocol-specific knowledge, the replay is performed by applying simple rules (such as replacing host names). In situations where the application requests data which was not recorded during the original session, the application can be allowed to go out onto the network or simply have its request denied (e.g. connection refused). Note that if the proxy is unable to provide the requested data, this is not a catastrophic problem because the lack of data itself signals a deviation from recorded behaviour. To account for the effects of non-determinism, applications are replayed several times in order to determine the average and the variance in the similarity of outputs. The outputs in question are the system calls made by the application.

It is obvious that the usefulness of network replay will be constrained by the lack of determinism (including timing issues), insufficient protocol knowledge, and situations where the requested data is unavailable, however we believe that for many replays it will prove useful (especially in cases where the behaviour is shown to be substantially similar with both versions of inputs). Some of the preliminary results found in [1], as a result of experiments performed by hand, are very encouraging. Our goal for this course project is to be able to replay some of the common Internet protocols such as HTTP.

RELATED WORK

There are many academic papers devoted to the concept of application replay, in this section, we narrow our discussion of the related work to the problem of network replay. The RolePlayer [2] uses heuristic methods to replay the client and server sides of network communications based on previous sessions and protocol-specific “hints”. Although we will be using several similar techniques to enable replay, we will not be attempting to dynamically determine the structure of unknown protocols. Nagaraja et al. [3] describe an approach based on replaying the requests of network clients and comparing the server’s responses similar to RolePlayer. Replayer [4] is a system for automatic protocol replay by binary analysis of the communicating parties. The solution it provides is sound but it is not always capable or cost effective to find a solution. Although not necessarily concerned with network replay, ReVirt [5] allows for deterministic replay of a system using a virtual machine monitor. This approach is more resource-intensive and it becomes more difficult to perturb a single input and to observe the behaviour of the application under replay.

REFERENCES

- [1] Fareha Shafique, Kenneth Po, Ashvin Goel. Correlating Multi-Session Attacks via Replay. To appear in the *Workshop on Hot Topics in System Dependability (HotDep)*, Nov 2006.
- [2] Weidong Cui, Vern Paxson, Nicholas Weaver, and Randy Katz. Protocol-independent adaptive replay of application dialog. In *Proceedings of the Network and Distributed System Security Symposium*, February 2006.
- [3] Kiran Nagaraja, Fábio Oliveira, Ricardo Bianchini, Richard P. Martin, and Thu D. Nguyen. Understanding and dealing with operator mistakes in internet services. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, pages 61–76, December 2004
- [4] James Newsome, David Brumley, Jason Franklin, Dawn Song. Replayer: Automatic Protocol Replay by Binary Analysis. In *Proceedings of 13th ACM Conference on Computer and Communications Security*, November 2006.
- [5] George W. Dunlap, Samuel T. King, Sukru Cinar, Murtaza Basrai, and Peter M. Chen. ReVirt: Enabling intrusion analysis through virtual-machine logging and replay. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, December 2002.